TerraWeather
Ensuring Your Safety & Success

AI Workshop – Lecture 6

Loss Functions

6.0 Point-wise Loss Functions

Most of the loss functions you have seen so far (and also in the literature) are "point-wise", meaning they operate on a single point in time of their input. Two commonly used point-wise loss functions are the Euclidean loss $(f(t)-\hat{f}(t))^2$ and Absolute loss $|f(t)-\hat{f}(t)|$.

Euclidean or other "point-wise" loss functions are often used to train and evaluate the performance of NNs. However, for time-series data, they present an incomplete picture, because they fail to capture the time aspect of the data.

To see this, in Figure 6.0, the (orange) predictions have a lower risk than persistence (grey) but visually and from an end-user's perspective, they are *qualitatively* the same – both are not much use.

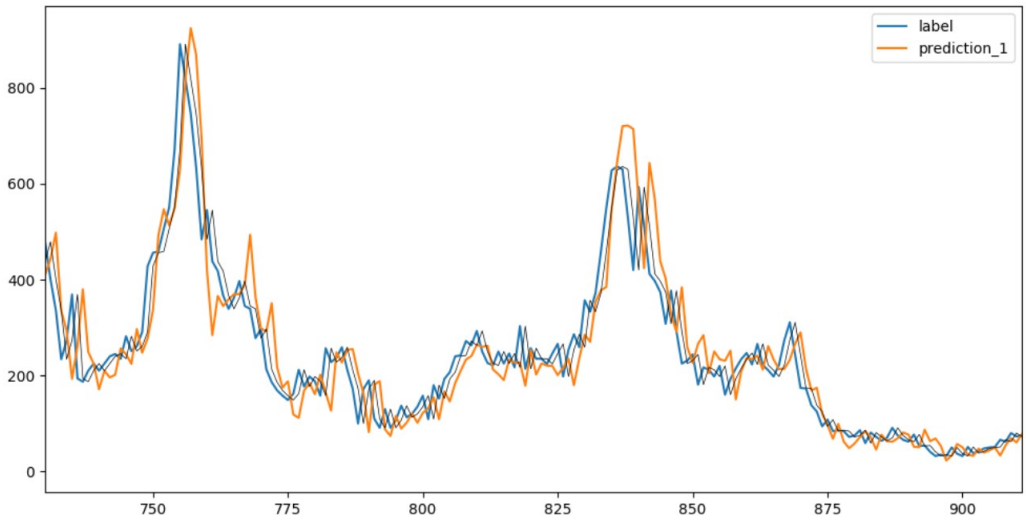

Figure 6.0 – A laggy prediction (orange), which is comparable to persistance (grey). Actuals are blue.

They are of not much use because both *lag* the labels, meaning that significant changes (like sharp peaks) in the labels precede the "predicted" peaks in both forecasts.

We will describe two very different approaches to this problem. In the first approach, we tackle lag directly.

6.1 The Problem of Lag

We can quantify lag by using the *correlation* function,

$$C(\Delta)=\sum_{k=1}^{N} f_k \hat{f}_{k+\Delta}$$

(Note that we implicitly set $\hat{f}_{k+\Delta}=0$ if $k+\Delta>N$ predictions or set then to cycle around to old values. Both choices are valid). We often find that the correlation between two similar series will result in a peak near $\Delta=0$. Figure 6.1 is a typical example.

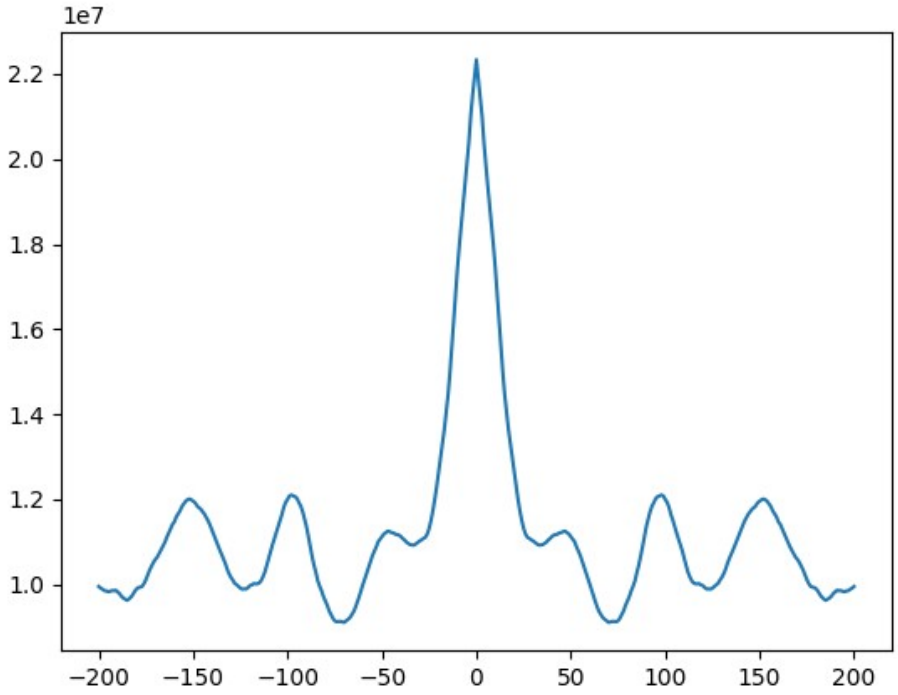

Figure 6.1 – A typical correlation plot. The horizontal axis is Δ . For this plot, the values were wrapped rather than being discarded.

If we define:

$$\hat{f}(t) = f(t+c)$$

then the correlation between $\hat{f}(t)$ and $f(t)$ will have a peak near $\Delta = c$ instead. This gives us a definition for lag:

$$lag[\boldsymbol{f}, \hat{\boldsymbol{f}}] = \underset{\Delta}{arg\,max}\, C(\Delta)$$

This definition means we find the offset Δ that gives the highest peak in *C*. Lag gives some idea of how much in advance/lagged our prediction is compared to actual events.

Quiz 6A: Attempt all questions.

1. Looking at the plot in Figure 6.1, what is the lag?

2. ** Is there a relationship between Risk induced by the Euclidean loss (or any other point loss) and lag? More specifically, if $R(f, f') < R(f, f'')$ does this imply that $lag(f, f') < lag(f, f'')$? Is the converse true? If "yes", sketch a proof. If "no" offer counterexamples.

3. Use Excel to plot the *autocorrelation* of a sine wave. The autocorrelation is the correlation of a function with itself.

4. Use Excel to plot the autocorrelation of a set of random numbers.

5. * Use Excel to plot the autocorrelation of dengue cases. You should see a single peak. What is the width of this peak? Do you think this has any significance?

6.2 Loss Functions for Time-series

As Quiz 6A.2 reveals, there is no relationship between the Risk functional induced by point-wise loss functions and the Lag. (You should offer examples).

This fact has a huge implication for time-series forecasting: simple point-losses under-constrain risk minimization, so that there is little connection between good training performance and lag during test.

What we need is a loss function that somehow penalized poor lag. This is illustrated in Figure 6.2:

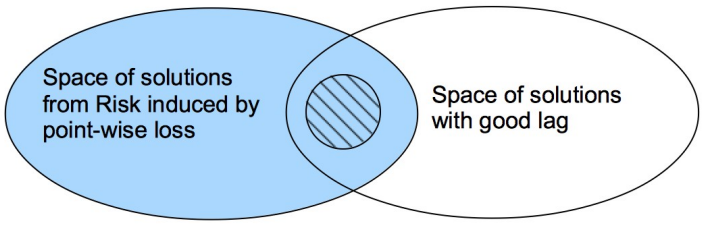

Figure 6.2 – The hatched area represents the space of solutions with low risk and good lag.

We can do this by adding to the loss function, “momentum” and “force” terms. These represent the local slope and curvature of the time-series labels.

As an illustration, consider the following Table 6.0 of Labels and Predictions:

k	Label	Prediction A	Prediction B
1	4	222	100
2	5	223	35
3	100	318	200
4	600	818	150
5	100	318	250
6	3	221	32
7	1	219	1
Euclidean Risk		**47,524**	**35,136**

Table 6.0: Labels vs. 2 predictions

Clearly, Prediction B is the “better” prediction based on the Euclidean Risk alone. However, a plot of the three (Figure 6.3) shows that this might not actually be the case.

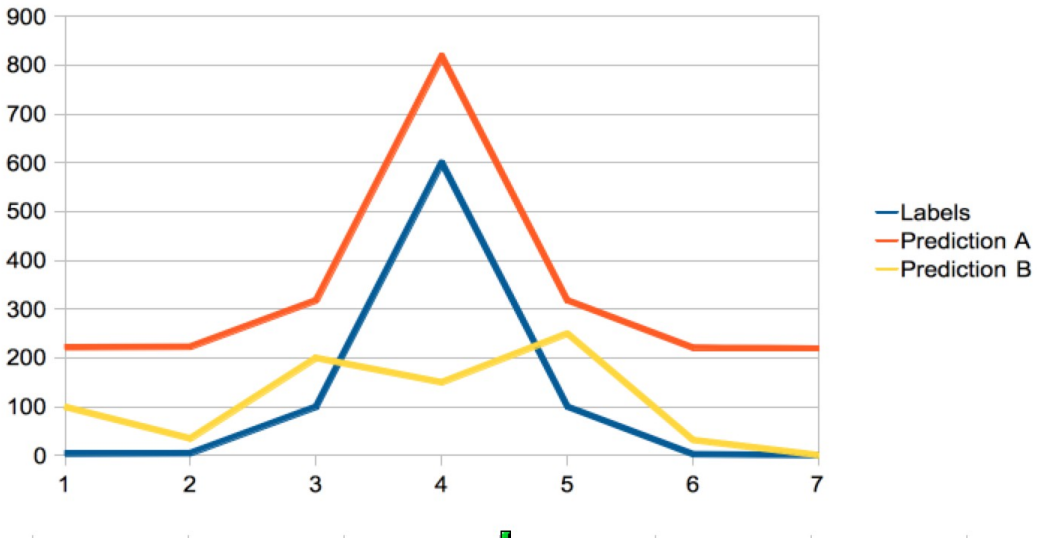

Figure 6.3 – Plots of Labels vs Prediction A and Prediction B. Which is "better" now?

Visually, Prediction A is much better, but the Euclidean Risk does not capture this. Unfortunately, your NN only "sees" the Risk functional. So, a badly crafted risk functional will always cause poor generalization.

The reason Prediction A "looks" better is because:

- The slopes of each section of Prediction A match that of the labels,
- The shape or curvature of Prediction A also matches that of the labels.

We can capture this intuition by defining the "Momentum" (M) and "Force" (F) of a time-series:

$$\begin{aligned} M_k &= x_k - x_{k-1} \\ F_k &= x_k - 2x_{k-1} + x_{k-2} \end{aligned}$$

Note that the momentum will shorten the time-series by 1 while the force will shorten it by 2. We apply these terms to the labels and predictions of the time-series, and treat them as separate labels/predictions of momentum and force.

The Risk is then adjusted to include the standard point-wise losses from these new series:

$$
\begin{aligned}
f_k &\rightarrow [f_k, M_k, F_k] \\
\hat{f}_k &\rightarrow [\hat{f}_k, \hat{M}_k, \hat{F}_k] \\
R(\boldsymbol{f}, \hat{\boldsymbol{f}}) &= E_D[\![L(f_k, \hat{f}_k) + L(M_k, \hat{M}_k) + L(F_k, \hat{F}_k)]\!]
\end{aligned}
$$

The additional loss terms can be quite big and swamp the original risk. Table 6.1 shows the individual Risk contributions using the data of Table 6.0:

	Prediction A	**Prediction B**
$E_D[\![L(f_k, \hat{f}_k)]\!]$	47,524	47,168
$E_D[\![L(M_k, \hat{M}_k)]\!]$	0	136,576
$E_D[\![L(F_k, \hat{F}_k)]\!]$	0	450,740
Total Risk	**47,524**	**634,485**

Table 6.1: A Comparison of Risk contributions

Using the new Risk, Prediction B is much worse compared to Prediction A, which is what we want.

Quiz 6B: (attempt all questions!)

1. Why is the plain risk $E_D[\![L(f_k, \hat{f}_k)]\!]$ for Prediction B in Table 6.1 different (higher) than that in Table 6.0?

2. * Use Excel to calculate the risk terms yourself, using the data of Table 6.0. Compare your answers to Table 6.1.

3. * Between Euclidean and Absolute loss, which do you think is more prone to problems with lag? Explain your answer clearly.

4. The new equation for Risk (including momentum and force terms) also uses the point-wise loss *L*. Does this contradict the statement in para 2 of Section 6.0?

5. Use Excel to calculate the total risk of a sine wave lagged by some adjustable parameter Δ. (The labels are a sine wave and the "predictions" are the lagged sine). Plot how the total risk varies with lag Δ

6. ** Revisit Quiz 6A.2: Does using the additional momentum & force contributions in risk necessarily imply lower lag? Defend your answer.

7. * What problems do you anticipate using the new momentum and force terms in risk? List these carefully.

6.4 Change the Problem

In many situations, you don't need to actually predict the time behaviour of a time-series. You can instead solve a different problem:

- Given the data we have now, predict the earliest time for event X. Predict its magnitude.

- Predict if within T timesteps, if the label will exceed a fixed threshold. Predict if the label will exceed X% of the current value.

These are very common problems which don't require you to explicitly forecast the time-series. In these situations you should:

1. **Decide what is an "event"**. This is not as trivial as it may seem! This definition of an "event" should be driven by your system's end-users, and be derivable from your data.

2. **Re-process your time-series data** to extract the necessary new labels relating to these events.

For example, with environmental data from Changi Airport, you might decide that a "squall" is a sustained, sudden increase of 10-metre, 1-minute averaged wind by 25 knots within a 15-minute time-frame. This definition would be driven by your end-users (presumably airport traffic controllers) and the availability of high-resolution environmental data (at least available every 15-minutes). Armed with this definition, you could either predict the presence or non-presence of a squall; or the time of arrival of the next squall.

This could be much simpler than predicting the wind time-series *and* easier for your model to find a good solution since you are focussing on the events relevant to your end-users.

Lab 6: Momentum & Force loss for dengue predictor

In this lab, you will

1. Modify the loss function by adding the momentum and force loss

Instructions

Dataset

A new dataset is provided in `~/caffe/data/dengue-2` . This dataset, like the `dengue` dataset, has size 20 window for each of the 3 input variables in the `data` dataset. The main difference is in the `label` dataset, which contains the full range of T+1 to T+20 labels at each time-step. This allows more flexibility in choosing the prediction horizon and simplifies the set up to find the momentum of the label. The dimensions of the datasets are given below.

`train.h5`

- `data`: (714 3 20)
- `label`: (714 20)

`test.h5`

- `data`: (178 3 20)
- `label`: (178 20)

Dataset configuration

Autocaffe has a feature that helps you change the dataset quite easily. Instead of having to create `train.txt`, `test.txt` and `val.txt` files for each experiment, all you have to do is add a line to `config.txt`

```
DATASET: dengue-2
```

When you do an `ac prep` command, autocaffe looks into the `~/caffe/data/dengue-2` directory for the pre-made `train.txt`, `test.txt` and (optionally) `val.txt` files, and copies them over to the results folder for your experiment. If the `val.txt` file does not exist, as in the case for the `dengue` and `dengue-2` datasets, then the `test.txt` will be used as `val.txt`, ie. in the results folder, `test.txt` and `val.txt` will have identical contents.

If you use this `DATASET` parameter, there is no need to have the three `.txt` files in your experiments folder any more.

The pre-made text files for dengue-2 dataset contain these paths

```
train.txt:
              /home/terra/caffe/data/dengue-2/train.h5

test.txt:
              /home/terra/caffe/data/dengue-2/train.h5
               /home/terra/caffe/data/dengue-2/test.h5

val.txt (copied from test.txt):
              /home/terra/caffe/data/dengue-2/train.h5
               /home/terra/caffe/data/dengue-2/test.h5
```

These paths were chosen to suit experiments using the normalization layer as most experiments will involve some form of normalization. If you want to exclude the train dataset during the test phase, you can use a Slice layer and slice along axis 0, index 714 to remove the train dataset and then Silence it.

Layer setup for **`train.prototxt`**
These are the layers needed and some of their key parameters. Make sure you know the purpose of each layer.

1. HDF5Data (`batch_size: 714`, train phase)
2. HDF5Data (`batch_size: 892`, test phase)
3. Norm (`bottom: "label"`)
4. Norm (`bottom: "data"`)
5. Power (`bottom: "normed_label"`, train phase) - given below
6. Slice (`bottom: "normed_label"`, test phase) - given below
7. Slice (`bottom: "sliced_label"`) - given below
8. Power (`bottom: "normed_data"`, train phase)
9. Slice (`bottom: "normed_data"`, test phase)
10. InnerProduct (`bottom: "sliced_data", axis: 1`)
11. TanH
12. InnerProduct (`axis: 1, num_output: 1`)
13. EuclideanLoss (`top: "loss"`)
14. Momentum and Force layers – refer to next section
15. Silence – given below
16. Silence (test phase)

Layers 5 and 6 are used to slice off the training dataset during the test phase but not during the train phase. Layer 5 is a Power layer without any specified parameters, which default to the identity operation. This layer thus just passes its input through to its output and is used for renaming the input blob during the train phase.

```
layer {
```

```
        name: "Power1"
        type: "Power"
        bottom: "normed_label"
        top: “sliced_label”
        include: {
                phase: TRAIN
        }
}

layer {
        name: "slice2"
        type: "Slice"
        bottom: "normed_label"
        top: "junk_0"
        top: "sliced_label"
        slice_param {
                axis: 0
                slice_point: 714
        }
        include: {
                phase: TEST
        }
}
```

The second Slice layer (layer 7) selects the prediction horizon for your prediction model. Here the T+16 label is chosen.

```
layer {
        name: "slice"
        type: "Slice"
        bottom: "sliced_label"
        top: "junk_1"
        top: "sliced2_label"
        top: "junk_2"
        slice_param {
                axis: 1
                slice_point: 15
                slice_point: 16
        }
}
```

Layers 8 and 9 have similar function to layers 5 and 6, but are applied to the normed data blob.

The unwanted blobs `junk_0` and `junk_1` contain the T+1 to T+15 and T+17 to T+20 labels respectively. They are fed to the Silence layer (layer 15) to be discarded.

```
layer {
	name: "Silence"
	type: "Silence"
	bottom: "junk_0"
	bottom: "junk_1"
}
```

Layers 6 and 9 produce their junk blobs only in the test phase, so a test-phase silence layer (layer 16) is needed to silence them.

Fill in the other layers, and refer to previous labs for examples.

Momentum

To recap the lecture, the momentum $M_t = x_t - x_{t-\Delta}$ where x_t is the prediction/label for time step t and Δ is a choosable spacing. Keeping in mind the need to calculate the force $F_t = x_t - 2x_{t-\Delta} + x_{t-2\Delta}$ later on, we will also extract from the next adjacent spacing $x_{t-2\Delta}$.

For this example, we apply $\Delta = 4$. Thus we need 6 time series: x_t, x_{t-4} and x_{t-8} for both prediction and label. The momentum are then found by subtracting accordingly.

`prediction` blob

Let's set the prediction blob (output of layer 12) as the x_{t-8} series. To make the x_{t-4} and x_t series, slice off the first 4 and 8 elements along axis 0, eg to get x_{t-4}

```
layer {
	name: "slicer"
	bottom: "prediction"
	slice_param {
		axis: 0
		slice_point: 4
	}
	top: "junk_3"
	top: "T4_left_truncated"
	type: "Slice"
}
```

Add another slice layer with `slice_point: 8` to get x_t.

Next we truncate the series from the other end to make x_t, x_{t-4} and x_{t-8} the same size. Slice both x_{t-4} and x_{t-8} at `slice_point: 706` (train set size – 8) during the training phase and `slice_point: 170` (test set size – 8) during

the test phase. The example below is for x_{t-4}

```
layer {
	name: "slicer"
	bottom: "T4_left_truncated"
	slice_param {
		axis: 0
		slice_point: 706
	}
	top: "T4"
	top: "junk_4"
	type: "Slice"
	include: {
		phase: TRAIN
	}
}

layer {
	name: "slicer"
	bottom: "T4_left_truncated"
	slice_param {
		axis: 0
		slice_point: 170
	}
	top: "T4"
	top: "junk_4"
	type: "Slice"
	include: {
		phase: TEST
	}
}
```

Remember to silence all the “junk” blobs by adding them as `bottoms` of the Silence layer.

Lastly use the Eltwise layer to do the element-wise subtraction:

```
layer {
	name: "subtract_1"
	type: "Eltwise"
	bottom: "T4"
	bottom: "T8"
	top: "mom_pred"
	eltwise_param {
		coeff: 1
		coeff: -1
	}
}
```

The eltwise layer defaults to performing an element-wise sum of its bottom blobs, each multiplied by its respectively `coeff` value. In the layer above the x_{t-4} blob (`"T4"`) is multiplied by 1 and the x_{t-8} blob (`"T8"`) is multiplied by -1 before both are added together to get the momentum $M_{\text{prediction}}$.

`label` blob

The label blob can be handled more easily as blob has the T+1 to T+20 labels along its axis 1, which we denote here as l_1 to l_{20} . First truncate the label blob along axis 0 to equalize the sizes of the prediction and label blobs.

```
layer {
	name: "slice_label"
	type: "Slice"
	bottom: "label"
	top: "junk1"
	top: "lbl"
	slice_param {
		axis: 0
		slice_point: 8
	}
}
```

The x_t value corresponds to l_T where T is the prediction horizon. Suppose the prediction horizon is 16 weeks. We then need to extract l_{16} , l_{12} and l_8 (which correspond to x_t , x_{t-4} and x_{t-8} respectively). To get l_8 , do another slice along axis 1:

```
layer {
	name: "L8"
	type: "Slice"
	bottom: "lbl"
	top: "junk2"
	top: "L8"
	top: "junk3"
	slice_param {
		axis: 1
		slice_point: 7
		slice_point: 8
	}
}
```

Do the same to get the l_{16} and l_{12} series, then subtract l_8 and l_{12} with an eltwise layer to get the momentum M_{label} .

For the momentum loss, use a Euclidean loss layer with $M_{\text{prediction}}$ and M_{label} as inputs. Silence the momentum loss so that the loss reported in the plots and stats functions are purely the loss between the prediction and label values.

```
layer {
       name: "mom_loss"
       bottom: "mom_pred"
       bottom: "mom_label"
       top: "junk4"
       type: "EuclideanLoss"
}
```

Remember to add another Silence layer for the additional junk blobs or shift the existing silence layer down and add the blobs to the layer.

Force
Finding the force loss involves subtracting the appropriate predictions and labels in $F_t = x_t - 2x_{t-4} + x_{t-8}$ and then compare them with a Euclidean loss layer (with a silenced output). This part is left as an exercise for you to fill in.

slice along axis 0, index 714 to remove the train dataset and then Silence it.

Layer setup for **`test.prototxt`**
The prediction network is simpler as the layers for slicing away the train dataset is not needed, as well as the momentum and force loss layers.

1. HDF5Data (`batch_size: 892`)
2. Norm (`bottom: "label"`)
3. Norm (`bottom: "data"`)
4. Slice (`bottom: "normed_label"`) - for selecting prediction horizon
5. InnerProduct (`bottom: "sliced_data", axis: 1`)
6. TanH
7. InnerProduct (`axis: 1, num_output: 1`)
8. Silence

The top blob of layer 4 is the label blob and has to have the string "`label`" in its name (eg, `normed_label`). Similarly the top blob of layer 7 has to be named `prediction_<prediction_horizon>`.

Test iterations
Since the batch sizes are equal to the dataset size, set the test iterations to 1 for both test phase and prediction run.

In `config.txt` (for prediction run):

```
TEST_ITERS: 1
```

In `solver.prototxt` (for test phase):

```
test_iter: 1
```

Homework

Set up experiments with and without the momentum and force loss. Be sure to use the repeat task to get the statistics of your results.

Report your best scores on our forum chat and **prepare for a short sharing of your results (10 mins + 10 mins Q&A)**. We will select a few groups to present at the next lecture.